TAKE-HOME ASSIGNMENT – Technical Lead (Aurore Labs)

Hola Emiliano,
Gracias por avanzar en el proceso para el rol de Technical Lead en Aurore Labs.

Nuestro objetivo es entender **cómo piensas, cómo estructuras sistemas, y cómo guiarías a un equipo junior**, más que evaluar perfección técnica.

Este ejercicio está diseñado para ser compatible con el uso de herramientas de IA.
Solo pedimos que **documentes tus decisiones y supuestos**.

PARTE A — Diseño y MVP de Sistema de Eventos (~45–60 min)

Aurore Labs necesita un **MVP funcional** para manejar eventos generados por múltiples aplicaciones internas (logs, métricas y eventos de usuario).

Cada evento tiene la siguiente estructura:

- timestamp
- service
- message
- metadata (opcional)

Requisitos del sistema:

El MVP debe cumplir con lo siguiente:

- **Ingestar hasta ~5.000 eventos por segundo** (no necesitas justificar números exactos; nos interesa tu razonamiento y dónde estarían los cuellos de botella).
- **Permitir búsquedas rápidas** por:
    - Rango de tiempo
    - Servicio
- **Retención automática de datos por 30 días**, sin intervención manual.
- **Exponer una interfaz de consulta** que pueda ser consumida por un panel interno (no es necesario implementar el panel; basta con la API o contrato).

- Ser un sistema económico, simple y mantenible, considerando que será operado y extendido por un equipo mayormente junior.

Tu tarea

Diseña y describe un MVP que cumpla estos requisitos. No pedimos perfección ni un sistema “enterprise”. **Evaluamos criterio, claridad y pragmatismo.**

Esperamos que cubras

1. Arquitectura general del MVP (alto nivel)
   - Componentes principales (idealmente 3–5).
   - Responsabilidad de cada componente.
   - Diagrama simple (ASCII, imagen o herramienta que prefieras).

2. Contrato y flujo básico
   - Cómo se ingesta un evento (por ejemplo: POST /events).
   - Cómo se consulta la información (por ejemplo: búsqueda por service y rango de tiempo).
   - Qué validaciones se hacen en el borde del sistema.

3. Código — Gestión de eventos (núcleo del MVP)

Escribe un **ejemplo de código o pseudocódigo** que represente cómo tu **MVP gestiona la ingesta de eventos.**

El objetivo es mostrar el **flujo y las decisiones,** no un sistema completo.

El código debe dejar claro:

- Cómo se reciben los eventos.
- Validación básica del payload.
- Buffering y/o batching.
- Cuándo y cómo los eventos se envían a almacenamiento.
- Qué ocurre cuando:
  - llega un evento inválido o corrupto,
  - el buffer se llena,
  - el sistema está bajo presión (por ejemplo, almacenamiento lento).

No es necesario usar frameworks ni una base de datos real.

4. Decisiones clave
   - Cómo manejas la ingesta (batching, buffering, backpressure).
   - Cómo y dónde almacenas los eventos, y por qué.
   - Cómo estructuras índices o particiones para optimizar consultas.
   - Cómo implementas la retención de 30 días.
   - Cómo evitas que errores puntuales rompan todo el pipeline.

5. Mantenibilidad para un equipo junior
   - Qué decisiones tomas para reducir complejidad.
   - Qué dejarías explícitamente fuera del MVP.
   - Qué documentación, estándares o guardrails definirías para el equipo.

6. Plan de implementación
   - 4–6 pasos para construir este MVP.
   - Qué partes delegarías a desarrolladores junior y cuáles tomarías tú como Tech Lead.

7. Futuras mejoras (bullet points)
   - Cómo evolucionaría el sistema si cambian:
     - el volumen de eventos,
     - los costos,
     - los requisitos de seguridad o compliance.

Notas importantes
- **NO pedimos código completo ni UI.**
- Puedes usar pseudocódigo, ejemplos simples o descripciones.
- Puedes usar herramientas de IA libremente, siempre que expliques tus decisiones.
- El foco está en cómo piensas y cómo diseñas un sistema operable, no en detalles de framework.

PARTE B — Mentoring y Code Review (~20–30 min)

Aquí tienes un código "escrito por un desarrollador junior".

```
# log_processor.py

import time
import json

class LogProcessor:
    def __init__(self, buffer=[]):
        self.buffer = buffer

    def process(self, log_event: dict):
        self.buffer.append(log_event)
        if len(self.buffer) > 100:
```

```
            self.flush()

    def flush(self):
        with open("logs.txt", "a") as f:
            for event in self.buffer:
                f.write(json.dumps(event))
        self.buffer.clear()

processor = LogProcessor()

# Example usage
processor.process({"service": "auth", "msg": "login", "ts":
time.time()})
```

Tu tarea

1. Identifica **problemas técnicos o de diseño**.
2. Explica **cómo se los comunicarías al junior de forma constructiva**.
3. Propón **mejoras simples** para que en el futuro trabajen mejor.

Evaluamos:

- tu claridad,
- tu capacidad de elevar al equipo,
- tu criterio para priorizar problemas.

PARTE C — Estimación y Planificación (~20 min)

Supón que el cliente pide:

> “Necesitamos tener la primera versión funcional del sistema (ingesta + consulta básica) en 6 semanas.”

Tu tarea

1. Da una estimación preliminar con:
    - supuestos

    - riesgos
    - dependencias
    - qué parte construirías tú vs. los juniors

2. Qué necesitarías para convertir tu estimación en un plan más preciso.

No buscamos exactitud absoluta, sino **cómo piensas y comunicas**.

Entrega

- PDF, Notion link o Google Doc (lo que te acomode).
- Puedes usar IA libremente, siempre que describas tus decisiones.
- Tiempo sugerido total: **1.5–2 horas**.